**Double Linked List**

*Source Code:*

#include <stdio.h>

#include <stdlib.h>

// Define a structure for a doubly linked list node

struct Node {

    int data;

    struct Node\* prev;

    struct Node\* next;

};

// Function to create a new node

struct Node\* createNode(int data) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    if (newNode == NULL) {

        printf("Memory allocation failed!\n");

        exit(1);

    }

    newNode->data = data;

    newNode->prev = NULL;

    newNode->next = NULL;

    return newNode;

}

// Function to insert a node at the beginning of the list

void insertAtBeginning(struct Node\*\* head, int data) {

    struct Node\* newNode = createNode(data);

    newNode->next = \*head;

    if (\*head != NULL)

        (\*head)->prev = newNode;

    \*head = newNode;

}

// Function to insert a node after a specific node

void insertAfterNode(struct Node\* prevNode, int data) {

    if (prevNode == NULL) {

        printf("Previous node cannot be NULL.\n");

        return;

    }

    struct Node\* newNode = createNode(data);

    newNode->next = prevNode->next;

    newNode->prev = prevNode;

    prevNode->next = newNode;

    if (newNode->next != NULL)

        newNode->next->prev = newNode;

}

// Function to insert a node at the end of the list

void insertAtEnd(struct Node\*\* head, int data) {

    struct Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

        return;

    }

    struct Node\* temp = \*head;

    while (temp->next != NULL)

        temp = temp->next;

    temp->next = newNode;

    newNode->prev = temp;

}

// Function to delete a node from the list

void deleteNode(struct Node\*\* head, struct Node\* nodeToDelete) {

    if (\*head == NULL || nodeToDelete == NULL)

        return;

    if (\*head == nodeToDelete)

        \*head = nodeToDelete->next;

    if (nodeToDelete->next != NULL)

        nodeToDelete->next->prev = nodeToDelete->prev;

    if (nodeToDelete->prev != NULL)

        nodeToDelete->prev->next = nodeToDelete->next;

    free(nodeToDelete);

}

// Function to display the doubly linked list

void displayList(struct Node\* head) {

    printf("Doubly Linked List: ");

    while (head != NULL) {

        printf("%d <-> ", head->data);

        head = head->next;

    }

    printf("NULL\n");

}

int main() {

    struct Node\* head = NULL; // Initialize an empty list

    int choice, data;

    char ch;

    do {

        printf("Doubly Linked List");

        printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert After a Node\n4. Delete a Node\n5. Display List\n6. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter data to insert at the beginning: ");

                scanf("%d", &data);

                insertAtBeginning(&head, data);

                break;

            case 2:

                printf("Enter data to insert at the end: ");

                scanf("%d", &data);

                insertAtEnd(&head, data);

                break;

            case 3:

                printf("Enter data to insert after: ");

                scanf("%d", &data);

                printf("Enter data to insert: ");

                scanf("%d", &data);

                insertAfterNode(head, data);

                break;

            case 4:

                printf("Enter data to delete: ");

                scanf("%d", &data);

                struct Node\* temp = head;

                while (temp != NULL && temp->data != data)

                    temp = temp->next;

                if (temp != NULL)

                    deleteNode(&head, temp);

                else

                    printf("Data not found in the list.\n");

                break;

            case 5:

                displayList(head);

                break;

            case 6:

                exit(0);

            default:

                printf("Invalid choice!\n");

        }

        printf("Do you want to continue (y/n)? ");

        scanf(" %c", &ch);

    } while (ch == 'y' || ch == 'Y');

    return 0;

}

*Output:*

Doubly Linked List

1. Insert at Beginning

2. Insert at End

3. Insert After a Node

4. Delete a Node

5. Display List

6. Exit

Enter your choice: 1

Enter data to insert at the beginning: 0

Do you want to continue (y/n)? y

Doubly Linked List

1. Insert at Beginning

2. Insert at End

3. Insert After a Node

4. Delete a Node

5. Display List

6. Exit

Enter your choice: 2

Enter data to insert at the end: 1

Do you want to continue (y/n)? y

Doubly Linked List

1. Insert at Beginning

2. Insert at End

3. Insert After a Node

4. Delete a Node

5. Display List

6. Exit

Enter your choice: 4

Enter data to delete: 0

Do you want to continue (y/n)? y

Doubly Linked List

1. Insert at Beginning

2. Insert at End

3. Insert After a Node

4. Delete a Node

5. Display List

6. Exit

Enter your choice: 5

Doubly Linked List: 1 <-> NULL

Do you want to continue (y/n)? y

Doubly Linked List

1. Insert at Beginning

2. Insert at End

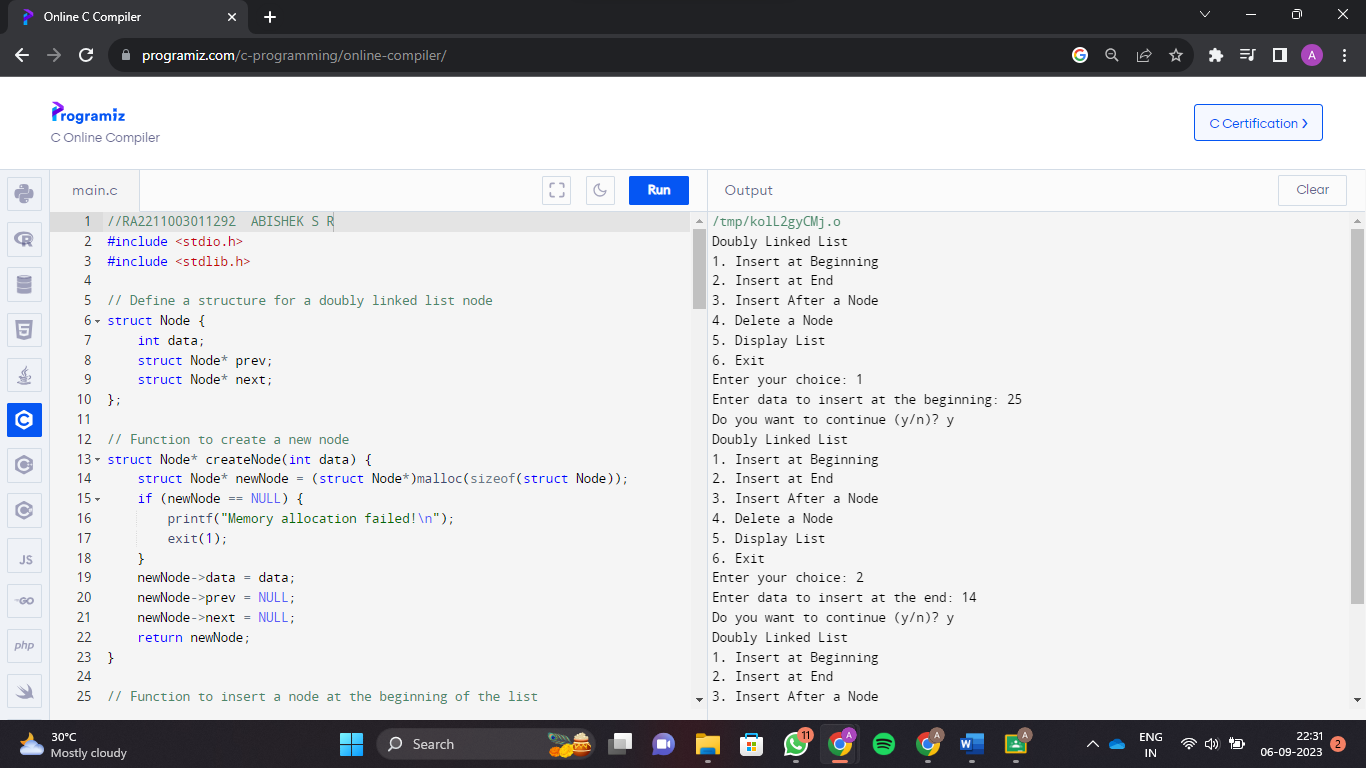
3. Insert After a Node

4. Delete a Node

5. Display List

6. Exit

Enter your choice: 6



**ABISHEK S R  
RA2211003011292**